Single Event Test Methodologies and System Error Rate Analysis for Triple Modular Redundant Field Programmable Gate Arrays

Gregory Allen, *Member, IEEE*, Larry D. Edmonds, Gary Swift, *Member, IEEE*, Carl Carmichael, Chen Wei Tseng, Kevin Heldt, Scott Arlo Anderson, and Michael Coe

***Abstract*—We present a test methodology for estimating system error rates of Field Programmable Gate Arrays (FPGAs) mitigated with Triple Modular Redundancy (TMR). The test methodology is founded in a mathematical model, which is also presented. Accelerator data from 90 nm Xilinx Military/Aerospace grade FPGA are shown to fit the model. Fault injection (FI) results are discussed and related to the test data. Design implementation and the corresponding impact of multiple bit upset (MBU) are also discussed.**

***Index Terms*—Error rate calculation, field programmable gate array, single event upset, triple modular redundancy.**

I. Introduction

VARIOUS methodologies for error detection and correction (EDAC) have been developed over the years to mitigate the system-level impact of device errors. The fundamental function of an EDAC mitigated system is to limit the effect of bit-errors at the system level. Devices hardened against single event upset (SEU) via EDAC require unconventional methods for estimating system error rates for space environments; simply integrating a cross-section versus linear energy transfer (LET) curve with an environmental LET spectrum is insufficient as the calculated system-error cross-section has a upset error rate dependence [1]–[4] (flux dependence if the LET is constant). The error rate dependence occurs in systems mitigated through redundancy due to the need for two upsets to occur in a given location in a set window of time. Hence, the greater the rate of upset, the higher the probability that the error will occur. This research specifically targets TMR-based EDAC mitigation, implemented within 90 nm Xilinx Static Random Access Memory (SRAM) based FPGAs. The approach that we present is to experimentally (via a particle accelerator) derive system error rates as a function of the underlying bit-flip rate. Theoretical models are presented, and compared to data obtained at flux levels practical for experimental work, that can be extrapolated to the lower flux levels experienced in space.

A complimentary approach to estimating system error rates discussed in this paper is fault injection. In FPGAs, fault injection is a technique whereby configuration bits are inverted to simulate an SEU, while test vectors are applied and the device output monitored. While fault injection is not comprehensive in the evaluation of failure modes of an FPGA, we have shown it to be very representative of the measured system error rates, as single points of failure in a design is an effective approximation to the overall system error rate. It should be noted that this methodology only applies to devices whose error rates are dominated by a single upset mode (in this case configuration upsets). This does not apply to devices where error rates are determined via various SEE, such as SEU in different resources, or a combination of SET and SEU.

TABLE I
XILINX VIRTEX-4 XQR4VLX200 FEATURE SET

	Description	XQR4VLX200
CFG*	Configuration Bits	43.0×10^6
BRAM	Block Memory Bits	6,193,152
LOGIC	Slices (2 Lookup Tables/slices)	89,088
DSP**	18x18 MACs	96
DCM	Digital Clock Managers	12
IOBs	Input/Output Blocks	960
CLB	Configuration Logic Block	50,112

* Only real memory cells in the Configuration Bit Stream are counted here (not counting BRAM)

** MAC=multiply-and-accumulate block for digital signal processing (DSP)

II. Background

In this section we provide a high-level device description in order to provide the reader with an architectural model to better understand the error rate model. We also provide an overview of the implemented designs and an abbreviated look into the error-rate model.

A. Device Description

For this testing we targeted a Xilinx XQR4VLX200, which is in the Virtex-4 family of 1.2 V SRAM-based FPGA. Table I below provides a list of the architecture resources. See [5]–[7] for detailed device description.

Manuscript received September 17, 2010; revised November 29, 2010; accepted December 28, 2010. Date of publication April 07, 2011; date of current version June 15, 2011.

G. Allen and L. Edmonds are with the Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA 91109 USA (e-mail: gregory.r.allen@jpl.nasa.gov).

G. Swift, C. Carmichael, and C. W. Tseng are with Xilinx Inc., San Jose, CA 80503 USA (e-mail: gswift@xilinx.com; carlc@xilinx.com; chenweit@xilinx.com).

K. Heldt, S. A. Anderson, and M. Coe are with SEAKR Engineering Inc., Centennial, CO 80111 USA.

Digital Object Identifier 10.1109/TNS.2011.2105282

Nuclear Science, IEEE Transactions on Volume: 58 , Issue: 3 , Part: 2
0018-9499/$26.00 © 2011 IEEE
Digital Object Identifier: 10.1109/TNS.2011.2105282
Publication Year: 2011 , Page(s): 1040 - 1046

TABLE II
XQR4VLX200 RESOURCE UTILIZATION

	Resource Utilization		
Design	**% Slices**	**% LUT**	**% FF**
Full TMR	37%	29%	10%
Partial TMR	41%	29%	15%

The following device architecture description corresponds to Virtex-4 family of FPGAs, but can generally be applied to all Virtex devices up to Virtex-4. The configuration logic block (CLB) is the fundamental component of the FPGA that provides function generators, registers, and routing controls. The CLB allows for implementation of macros or other functions. Each CLB is connected to a switch matrix and then to the general routing matrix. A basic understanding of the CLB and the implementation options surrounding it is imperative in mitigating various SEE responses. There are four slices in any given CLB and each slice contains 2 look-up tables (LUT), 2 registers [flip-flops (FF) or latches], and carry logic. Two of the 4 slices in each CLB can be configured into 64 bits of distributed RAM (LUTRAM) or 64 bits of shift registers (SRL16). Each LUT can be configured into an arbitrary, user-defined, 4-input Boolean function. The registers can either be configured as edge-triggered D-type flip-flops or level sensitive latches. The control signals include clock (CLK), clock enable (CE), and set/reset (SR). The control signals of the registers, along with control signals of other primitives in the device, are provided logic levels with weak keeper pull-ups and pull-downs known as half-latches. Half-latches are fixed within the FPGA, i.e. they are not controlled by programmable bits and therefore cannot be scrubbed, but are routed to the correct location. The weak resistive ties can be overridden by any hard routing signal. The functionality of all of the architectural components are defined and/or initialized by configuration bits. SEU of the configuration bit is the dominating error mode from a system perspective. That is to say, SEU to configuration bits dominate system-level failures relative to upsets to flip-flops, BRAMs, LUTs, etc.

B. Design Description

Two sets of designs were studied. The first was a scalable "counter" design and the second was a digital clock manager (DCM) mitigation design. All designs used the Xilinx TMRTool [8], [9] to automate the triplication process. The counter design was implemented both as a fully mitigated and partially mitigated design. Table II provides an overview of the design's resource utilization.

The partially mitigated design left certain design elements unmitigated (untriplicated). In each of the counter designs, a module of 13 8-bit counters was implemented as the base building block and scaled to fill the design. Single, triplicated clocks and reset lines were provided for the scaled counter designs. Unique initialization values were used for each of the 8-bit counters.

The mitigated DCM design consisted of three separate DCMs (such that a single mitigated DCM block consists of three individual DCMs), each in their own clock domain with no clock voting taking place (although control logic was triplicated). The design was implemented with GLUTMASK disabled, i.e. the DCM's dynamic reconfiguration port (DRP) associated bits were being scrubbed. Without GLUTMASK disabled, the DRP bits would not be scrubbed. Each output clock of each individual DCM fed a two-bit counter, whose output went into a minority voter. If a discrepancy occurred in the output counters, the DCM was reset. A system failure was counted if two or more DCMs needed to be reset at the same time, where "same time" is defined as happening within a single scrub cycle. All counter outputs and reset counts were monitored and recorded in a strip-chart during the tests.

C. Model Description

Before discussing how to predict and measure error rates on TMR'ed systems, a set of terminology must be established. As discussed in detail in [8], for every feedback path or register, a triplicated set of majority voters is inserted. This grouping of triplicated logic, registers, and voters is referred to as Groups, and the number of them is denoted in the following equations as M. The scrub time, T_C, is the period by which the device is monitored and scrubbed for SEU. The underlying per-bit upset rate (bit-errors per bit-second), r, is a product of two measurement parameters: the per-bit configuration SEU cross-section $[\sigma_{per-bit}(L)]$ and the ion flux, f. A per-bit cross section is calculated by dividing static upsets by fluence, and is assigned to the effective LET of the test particle selected by the test engineer (see Section *Static SEU Response* in Section IV for example). Typically, once a convenient effective LET is selected, only the flux $(\mathrm{ions/cm^2/sec})$ is varied to achieve data that is plotted against the theoretical prediction. While either parameter can be varied to achieve various values of r, the flux is generally varied to save time at the accelerator. For each run, the system-error cross-section (per device) is calculated in the usual manner (the number of system errors divided by the total fluence). The system-error cross-section is multiplied by the recorded flux and the system-error rate (system errors/sec), R, is the product. The system-error rate is then plotted against the raw bit-flip rate showing an experimental determination of R as a function of r.

The theoretical determination of R and its approximation [(4)], which is derived in [3], is given by

$$R = \frac{1}{T_C} - \frac{1}{T_C}\prod_{i=1}^{M}\left[3\exp(-2N_i r T_C) - 2\exp(-3N_i r T_C)\right]. \tag{1}$$

where N_i represents the number of configuration bits in a single triplicated domain of the ith group. The exact (1) can be approximated as

$$R \approx \frac{1}{T_C}\left\{1 - \exp\left[-3M(\mathcal{M}_2 r T_C)^2 + 5M(\mathcal{M}_3 r T_C)^2 - \frac{37}{4}M(\mathcal{M}_4 r T_C)^4\right]\right\}. \tag{2}$$

where $\mathcal{M}_2$, $\mathcal{M}_3$, and $\mathcal{M}_4$ are three moments instead of the complete set of M moments. These three are defined by

$$\mathcal{M}_2 \equiv \left[\frac{1}{M}\sum_{i=1}^{M} N_i^2\right]^{1/2}, \quad \mathcal{M}_3 \equiv \left[\frac{1}{M}\sum_{i=1}^{M} N_i^3\right]^{1/3},$$
$$\mathcal{M}_4 \equiv \left[\frac{1}{M}\sum_{i=1}^{M} N_i^4\right]^{1/4}.$$

In practice, $\mathcal{M}_2$, $\mathcal{M}_3$, and $\mathcal{M}_4$ typically have the same values. Approximating the number of bits (N_i) in a triplicated domain is very impractical. In contrast, the number of groups, M, is fairly easy to estimate using a tool such as FPGA editor that can count the number of instantiations given by a name, and T_C is typically well defined by the designer.

The above approximation ((2)) was augmented to fit data that contained single points of failure, i.e. unmitigated circuitry. The augmented approximation, derived in the Appendix, is given as,

$$R' = \frac{1 - e^{-M_U r_U T_C}}{T_C} + e^{-M_U r_U T_C} R \tag{3}$$

where M_U is the total number of unmitigated configuration bits, and R is calculated from (2) for the mitigated bits. An approximation to (2) was derived and described as

$$R \approx 3MT_C(\mathcal{M}_2 r)^2 \quad (\text{small} - r). \tag{4}$$

III. Experimental Procedure

A. Facilities

The DCM testing was performed at Lawrence Berkeley National Laboratory's (LBNL) 88-inch cyclotron in vacuum. A 16 MeV per AMU Argon beam was used with an effective LET of 7.27 $\mathrm{MeV} - \mathrm{cm}^2/\mathrm{mg}$. The range of this beam is 256 um in silicon. The mitigated counter testing was performed at the Texas A&M Cyclotron in air. A 24.8 MeV per AMU Argon and Krypton beam were used to provide an effective LETs of 6.4 and 27.9 $\mathrm{MeV} - \mathrm{cm}^2/\mathrm{mg}$ respectively. The backside of the test devices were thinned to a nominal thickness of 100 $\mu\mathrm{m}$.

B. Accelerator Test Methodology

Prior to performing mitigated testing, the underlying upset rate must be established as discussed in [5]. When testing a mitigated design, a convenient LET is selected, where "convenient" means that adequate counting statistics can be obtained from flux levels available at the accelerator and using reasonable beam run times. It is important to note that the LET selected should, if possible, be in the saturated region of the $\sigma_{per-bit}(L)$ curve. If it is selected in the knee or below, slight variations in LET measurements will greatly affect r. In this set of tests, the flux f is varied from one run to the next. For each run, the system error cross section (per device and defined in the usual way, i.e., counts divided by fluence), denoted $\sigma_{SYS}(f)$, is measured and the flux is recorded. This produces a plot of the system error cross section as a function of flux. This plot is then converted into a plot of R versus r by multiplying the vertical coordinate $\sigma_{SYS}(f)$ by f to obtain R, and by multiplying the horizontal coordinate f by $\sigma_{per-bit}(L)$ (where L is the LET used during the second set of tests) to obtain r. The final result is an experimental determination of R as a function of r.

C. Fault Injection Test Methodology

Partial reconfiguration fault injection is performed in a similar manner to a configuration readback-scrubbing routine. When this approach is applied, a configuration controller dynamically and partially reconfigures the configuration bitstream through the SMAP or JTAG port. While it is possible to perform dynamic partial reconfiguration through programming tools such as iMPACT, similarly to full reconfiguration, it is a very manual practice, and again, full coverage is impractical to accomplish.

One method to attain full coverage is accomplished through iterative single-bit, partial reconfiguration fault injection; such was the method used in this work. The Xilinx Radiation Test Consortium's fault injection system employs an automated configuration controller (CONFIGMON) and functional monitor (FUNCMON) that handshake with each other. The state flow for verifying a design is as follows:

Upon power up, CONFIGMON is initialized and configures the DUT. It is then in an "IDLE" state until a command is given. Once the command to inject faults is given, a corrupted frame is written to the DUT. Anywhere between 1 and 8 bits can be corrupted in a single frame during any given write. Once the write has been confirmed, CONFIGMON will request FUNCMON to run through a user-defined set of test vectors to detect whether or not the fault injection caused a system error. Once the test vectors have fully exercised the DUT, FUNCMON reports that is has completed its test and reports the system status (system failure or not). If FUNCMON reports a system failure, the address and bit(s) of the injected fault(s) are recorded in a register that is strip charted. For every observed error, recovery methods such as system reset, configuration scrub, or full reconfiguration are attempted sequentially and recorded until full functionality is regained. If no system error is reported no action is required. The corrupted bit within the frame is then scrubbed (corrected) then the next bit within that frame is corrupted. Once all bits within the frame have been corrupted, the frame address register (FAR) is incremented, and process repeats. The fault injection core has the capability of performing the fault injection with and without SEFI detection. Upon SEFI detection, the user has the option to have an automated recovery (automatically record the SEFI and pulse PROG to reconfigure) or to pause, remain in the IDLE state, and manually reconfigure the device. This process is repeated for the entire bitstream.

Depending upon the size of the device, set of functional test vectors, and speed setting of the fault injector, this process may take minutes to weeks to complete. However, the resultant number of single points of failure is used to achieve an effective approximation to the system error rate of the device.

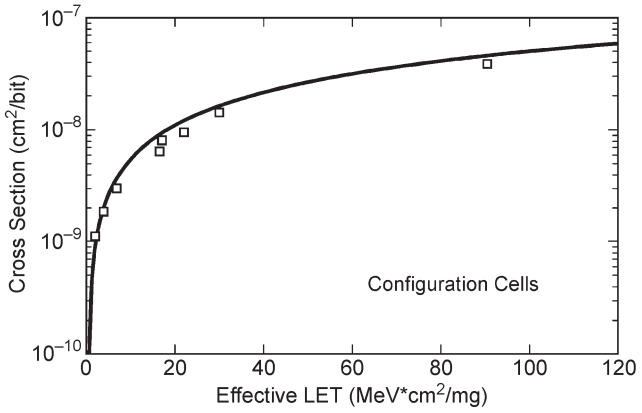

Fig. 1. Static SEU per-bit cross-section versus effective LET for an XQR4VLX200's configuration cells [5].

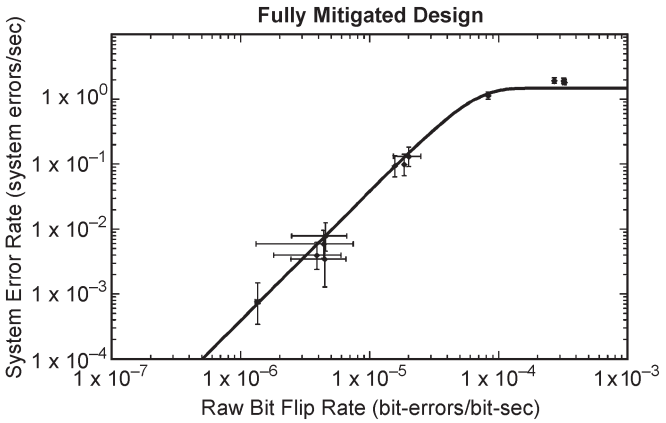

Fig. 2. Experimental data shown with the theoretical model (2) for R versus r of the fully mitigated design. Vertical error bars represent the Poisson distributed statistical error in counting system error events. The horizontal error bars represent recorded fluctuations in the ion beam flux when recorded (instantaneous flux was not recorded for every run).

IV. Test Results and Discussion

A. Static SEU Response

The static SEU response is shown in Fig. 1. A full description of how the data was acquired and fitting parameters is available in [5].

B. Fault Injection Results

Fault injection was performed on each design, and after several iterations, it was found that there were no single points of failure in the fully mitigated counter design, nor the mitigated DCM design. The partially mitigated counter design exhibited 12 fault injection points that recovered via reset alone, and 4016 that recovered via a reset and scrub.

The iterative process of triplication and fault injection exemplified how automated triplication isn't a push-button process. This form of mitigation requires in depth mitigation verification. To achieve full triplication, it is necessary to apply area constraints to the triplicated domains, else routing bits will become single points of failure due to domain crossing. Area constraints physically separate user-defined portions of a design. In this instance, we applied the area constraints to the three triplicated domains, physically separating them on the chip to minimize the likelihood of the place and route tools implementing the design in such a way that domain crossing occurs. However, this can severely limit the overall size and speed of the design. Tradeoffs between performance and reliability must be quantified and decided upon. The application of area constraints also reduces the probability of MBU inducing system-level failures.

C. Mitigated Results, Rate Analysis, and Implications of Test Results

For both of the counter designs, T_C was engineered to be 0.669 seconds. M was estimated to be 11440 for the fully mitigated design, and 8650 for the partially mitigated design through the use of the Xilinx PlanAhead Tool [10], which provides resource utilization statistics including voters. From this data, M is determined. M_U was measured to be 4016 for the partially mitigated design from the fault injection results. Figs. 2 and 3 show accelerator data (upper curve in Fig. 3) and the theoretical

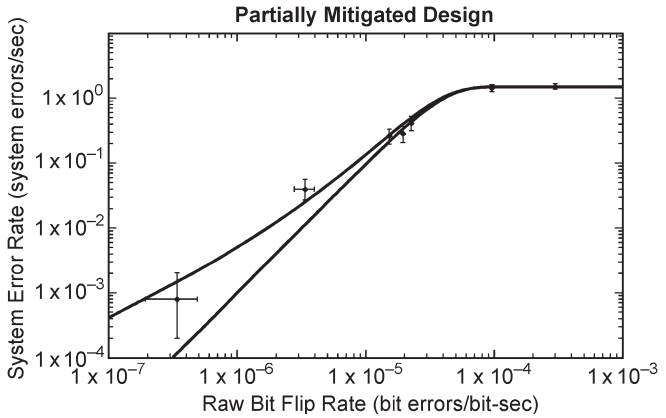

Fig. 3. Experimental data and fit (upper fit) for the partially mitigated counter design shown with the theoretical model (3) for R' versus r of the partially mitigated design. Vertical error bars represent the Poisson distributed statistical error in counting system error events. The horizontal error bars represent recorded fluctuations in the ion beam flux when recorded (instantaneous flux was not recorded for every run).

fit (lower curve in Fig. 3) for the fully and partially mitigated designs, respectively. The theoretical curve in Fig. 3 represents a perfectly TMR'ed design, (2), which we do not have. The upper fit corresponds to (3), where single points of failure have been taken into account.

The remaining variables in the theoretical fit, $\mathcal{M}_2$, $\mathcal{M}_3$, and $\mathcal{M}_4$ (the three moments of the N_i quantities), are defined by fitting the model to the experimental data. The moments were found to be 122 and 302 for the fully and partially mitigated designs respectively. Once an appropriate fit is defined, rates in a given space environment should be estimated by first estimating the expected raw bit-flip rate for that environment. Then, the appropriate fit (e.g., the fit used to produce the curve in Fig. 3 for the fully mitigated design) is used to estimate the system error rate. For example, the expected raw bit flip rate for a Virtex-4 device due to GCR in solar minimum conditions is 3.2×10^{-12} bit − errors/bit − second [5]. When that is applied to the fully mitigated design, we can expect an error rate of 4.0×10^{-15} system errors/second, or 1.3×10^{-4}

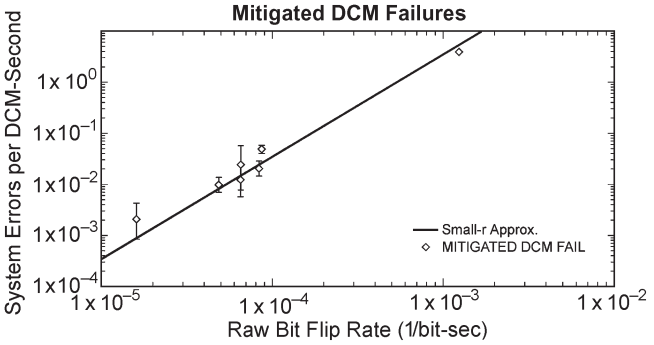

Fig. 4. Experimental data shown with the small-r approximation (4) for R versus r of the partially mitigated design. Vertical error bars represent the Poisson distributed statistical error in counting system error events.

system errors/millennium. When applied to the partially mitigated design, we can expect an error rate of 1.3×10^{-8} system-errors/second, or approximately one event every 2.5 years. The failure rate for the partially mitigated design is the same as the expected upset rate for 4016 configuration bits, the number of single points of failure in the design (i.e. $3.20 \times 10^{-12}\ \mathrm{bit-errors/bit-second}$ times 4016 bits equates to $1.3 \times 10^{-8}\ \mathrm{bit-errors/second}$). What this shows us is that if properly implemented, fault injection can be a very accurate tool to predict system error rates. However, the accelerator will always produce the most exhaustive verification. Furthermore, the model can also be used to predict system error rates from high-flux events such as solar flares.

Other predictions can also be made. Once accelerator data is acquired to verify accuracy of the R-model, the design engineers can now alter scrub times to see how that will affect the failure rate. For example, instead of scrubbing the fully mitigated design at $\mathrm{TC} = 0.669\ \mathrm{seconds}$, we implement a scheme where the configuration manager blindly reconfigures the part once per day. The predicted system error rate now becomes 5.1×10^{-10}, or a little more than 1.6 system errors per century (on the order of the device SEFI rate). It should be noted that if the same reconfiguration methodology is applied to the partially mitigated design, it does not affect the failure rate, as the single points of failure dominate the system error rate. However, the scrub time will affect the recovery rate, i.e., unless the configuration manager is alerted to the functional failure, the system will not recover until the blind reconfiguration is executed.

The fit shown in Fig. 4 shows a quadratic relationship between the system error rate for DCMs and the raw bit flip rate, as we expect to see. The small-r approximation was used to fit this data, and as such is only applicable for small r, i.e. the fit does not saturate. The result reinforces the belief that the design is correctly TMR'ed (as implied from the fault injection testing).

A space rate for this design can now be extrapolated by extending this trend down to lower flux levels encountered in space. Using the same r (GCR, solar minimum) as in the previous examples and extrapolating the quadratic fit from Fig. 4 to the expected bit flip rate, we get a failure rate of approximately 1.0×10^{-2} system errors per DCM circuit-century, i.e. once every 10,000 years. This rate compared to the device SEFI rate, which for a GEO orbit is approximately one per century, implies a more than satisfactory rate for this mitigation strategy. The device SEFI error rate is independent of design, and cannot be mitigated out. The SEFI error rate is the best system error rate one can expect to achieve from these devices, so once a design has been mitigated below that rate, the device will have achieved maximum robustness.

It should be noted that all testing was performed at normal incidence. While no MBU effects were observed in the data (namely due to the area constraints applied to the designs), the probability of MBU induced system failures will increase at angle. This effect was not quantified in this study, and is a possible area of future work. MBU effects will manifest within the data as single points of failure. That is to say, at low bit-flip rates, MBU induced system errors will begin to show the same deviation from the model as single points of failure.

V. Summary

In summary, we have presented a model and test methodology for determining error rates of TMR'ed systems. We have shown that the model fits accelerator data and correlates to bench-top fault injection measurements.

Appendix

1) System Error Rates for Devices Having Partial TMR Protection: A previous analysis [3] calculated system error rates produced by single-event upsets in triple-modular redundancy (TMR) devices. The previous analysis considered devices that are fully protected by TMR. Here we consider devices in which one portion is protected by TMR while another portion is not. A portion of the device (e.g., a memory array) is fully protected by TMR, while another portion (e.g., an output buffer) has no TMR protection. The protected portion is called "System-P" and the unprotected portion is called "System-U" in the figure. There is an infinite variety of ways in which a device can be partially protected by TMR, but selecting a particular prototype for illustration makes visualization easier. The entire device is a black box in the sense that the only thing that is observable is the signal at the output. The outside world cannot distinguish an error in System-P from an error in System-U because either one will produce an error at the output. There is an internal distinction, which affects the probability that an error will occur, because an error in System-P is a TMR error as defined in [3], while an error in System-U is one or more of any type of single-event effect (e.g., bit flips, resets, SEFI) that have no TMR mitigation. For the prototype selected for discussion, System-U is susceptible to bit flips, and any bit flip in this system at any time will produce an error at the device output.

As previously stated, a system error is either a TMR error in System-P, or a bit flip (one or more) in System-U, or both. Even with a system error already defined, the system error rate still requires a definition because it depends on the counting convention. We will follow the same convention used in [3], so we start with a review of the convention used there. This review focuses on System-P. The counting convention used in [3] for System-P was motivated by the assumption that the device output is checked for errors at the end of each TMR cycle. Errors in multiple TMR groups during the same cycle are seen at

the device output as one system error, so there can be at most one System-P error per cycle. The error rate (denoted R_P for System-P) is defined to be the expected number of errors (denoted N_P for System-P) during one cycle divided by the cycle duration T_C, i.e.,

$$R_P \equiv \frac{N_P}{T_C}.$$

The expected number N_P follows the counting convention that the number of System-P errors during one cycle is either 0 or 1, so the expected number is the same as the probability (denoted P_P for System-P) of an error during one cycle. This gives

$$R_P = \frac{P_P}{T_C}. \tag{A1}$$

Now consider the composite system consisting of System-P and System-U taken together. The error rate for the composite system, denoted R, is defined by

$$R \equiv \frac{N}{T_C}$$

where N is the expected number of errors during one cycle for the composite system. Multiple bit errors in System-U during the same cycle are seen at the device output at the end of the cycle as one output error, so there can be at most one system error per cycle for the composite system. The number of system errors for the composite system during one cycle is either 0 or 1, so the expected number is the same as the probability, denoted P, of an error in the composite system during one cycle. This gives

$$R = \frac{P}{T_C}. \tag{A2}$$

To continue with the analysis, we need to evaluate the probability P appearing in (A2). The probability of a System-P error during one cycle was already denoted P_P. Let P_U denote the probability of a System-U error during one cycle, which is the probability of one or more bit flips in System-U during one cycle. The probability function is an additive set function and has the property that can be written in generic notation as

$$P(A \cup B) = P(A) + P(B) - P(A \cap B)$$

where A and B are arbitrary events (subsets of the sample space). If the events are statistically independent we also have $P(A \cap B) = P(A)P(B)$ and the above equation becomes

$$P(A \cup B) = P(A) + P(B) - P(A)P(B).$$

In particular, let the event A be a System-P error and let the event B be a System-U error. These events are statistically independent so the above equation applies and gives

$$P = P_P + P_U - P_P P_U = P_U + (1 - P_U)P_P.$$

Substituting this into (2) and using (1) gives

$$R = \frac{P_U}{T_C} + (1 - P_U)R_P. \tag{A3}$$

To finish the analysis we need to evaluate P_U and R_P appearing in (A3). Recall that P_U is the probability of one or more bit flips in System-U during one cycle. Poisson statistics gives

$$P_U = 1 - e^{-M_U r_U T_C}$$

where M_U is the number of bits in System-U and r_U is the per-bit bit-flip rate for the bits in System-U. Substituting this result into (A3) gives

$$R = \frac{1 - e^{-M_U r_U T_C}}{T_C} + e^{-M_U r_U T_C} R_P. \tag{A4}$$

Finally, an expression for R_P was derived in [3] and is given by

$$R_P = \frac{1}{T_C}\left\{1 - \exp\left[-3M(\mathcal{M}_2 r_P T_C)^2 + 5M(\mathcal{M}_3 r_P T_C)^3 - \frac{37}{4}M(\mathcal{M}_4 r_P T_C)^4\right]\right\} \tag{A5}$$

where r_P is the raw (i.e., with TMR disabled) per-bit bit-flip rate for the bits in System-P, and the parameters M, M_2, M_3, and M_4 are defined in [3]. Note that (A5) is actually an approximation, but the accuracy is so high that it is considered exact in the discussions given here.

The final result consists of (A4) and (A5). This result simplifies if r_U and r_P are both sufficiently small because (A5) reduces to

$$R_P \approx 3MT_C(\mathcal{M}_2 r_P)^2 \quad (\text{small } r_P)$$

while the exponential function in (A4) can also be approximated. The result is

$$R \approx M_U r_U + 3MT_C(\mathcal{M}_2 r_P)^2 \quad (\text{small } r_P \text{ and } r_U). \tag{A6}$$

If the bits are the same in both subsystems, so that r_U and r_P have a common value that can be denoted r, then a plot of R versus r should follow the quadratic form given by

$$R \approx M_U r + 3MT_C(\mathcal{M}_2 r)^3 \quad (\text{small } r)$$

until r becomes large enough to make it necessary to replace (A6) with (A4) and (A5).

REFERENCES

[1] L. Edmonds, A Basic Analysis of EDAC Word Error Rates Internal JPL Interoffice Memo, Jun. 23, 2000.

[2] J. A. Zoutendyk *et al.*, "Single-Event Upset (SEU) in a DRAM with on-chip error correction," *IEEE Trans. Nucl. Sci.*, vol. NS-34, no. 6, pp. 1310–1315, Dec. 1987.

[3] L. Edmonds, Analysis of SEU Rates in TMR Devices Internal Document, Nov. 22, 2008 [Online]. Available: http://trs-new.jpl.nasa.gov/dspace/bitstream/2014/41123/1/09-6.pdf

[4] D. G. Mavis *et al.*, "Multiple bit upsets and error mitigation in ultra-deep submicron SRAMS," *IEEE Trans. Nucl. Sci.*, vol. 55, no. 6, pp. 3288–3294, Dec. 2008.

[5] G. R. Allen, G. Swift, and C. Carmichael, Virtex-4QV Static SEU Characterization Summary [Online]. Available: http://parts.jpl.nasa.gov/docs/NEPP07/NEPP07FPGAv4Static.pdf

[6] G. R. Allen, Virtex-4QV Dynamic and Mitigated Single Event Upset Characterization Summary [Online]. Available: http://trs-new.jpl.nasa.gov/dspace/bitstream/2014/41104/1/09-04.pdf

[7] Radiation Tolerant Virtex-4 QPro Family Overview, Dec. 16, 2008 [Online]. Available: http://www.xilinx.com/support/documentation/data_sheets/ds653.pdf

[8] C. Carmichael, Triple Module Redundancy Design Techniques for Virtex FPGAs [Online]. Available: http://www.xilinx.com/support/documentation/application_notes/xapp197.pdf

[9] *"Xilinx TMRTool User Guide,"* ver. UG156 v2.2, Sep. 12, 2007.

[10] *"PlanAhead User Guide"* ver. UG632, Dec. 1, 2009 [Online]. Available: http://www.xilinx.com/support/documentation/sw_manuals/xilinx11/PlanAhead_UserGuide.pdf, accessed Nov. 29, 2010